COSMIC Functional Sizing based on UML Sequence Diagrams

Thomas M. Fehlmann, Eberhard Kranich

Euro Project Office AG, Zurich, Switzerland
T-Systems International GmbH, Bonn, Germany

thomas.fehlmann@e-p-o.com, eberhard.kranich@t-systems.com

Abstract:

Agile Teams use UML sequence diagramming to communicate their designs with customers and sponsors. It is tempting to count functional size directly from UML sequence diagrams, once these are available. However, what are the caveats and risks?

To understand this, the authors have made a fictional case exercise, looking at a typical software problem of today involving web services and social media integration. It turns out that the COSMIC Measurement Method – although ten years old – provides not only functional sizing but encourages good software engineering practices such as sequence diagramming.

This paper provides an overview of what has been done so far and presents and reviews experiences made with the COSMIC methodology (ISO/IEC 19761:2011) in the COSMIC community when counting sequence diagrams.

Keywords

COSMIC Functional Sizing, Sequence Diagramming, Agile Software Development, Requirements Elicitation, Software Architecture.

1 The Modern Art of Developing Software

The paradigm inherited from traditional software projects – count size initially, possibly mid-term, and at the end of development – is not suitable for agile development since each time a software artefact has been completed functional size measurements have to be updated and actualised, see [17]. Multi-layered software works in systems involving people, devices, networks, and computers and meets high levels of security standards.

Developing software is iterative or even agile - in both cases it is hard to predict how a system looks when it is completed because the system's (commercial) success depends heavily from more or less quickly changing market demands. Developing according a *plan* is becoming the exception – usually, a *vision* must do. The requirements grow as the system is growing, and during development they are hence better and better understood by its stakeholders.

1.1 Sizing with the International Standard ISO/IEC 19761 "COSMIC"

The international standard ISO/IEC 19761 defines how to decompose a system into layers, and size layered software based on a generic software model identifying *Functional Processes, Data Groups,* and *Data Movements* in response to *Functional User Requirement* (FUR) [1]. Data movements are classified into the following four sub-types:

- An *Entry* (E) is a data movement that moves a data group from a functional user across the boundary of the respective application into the functional process where it is required. An Entry is considered to include certain associated data manipulations.
- An *Exit* (X) is a data movement that moves a data group from a functional process across the boundary of the respective application to the functional user that requires it. An Exit is considered to include certain associated data manipulations (e.g., formatting and routing associated with the data to be exited).
- A *Read* (R) is a data movement that moves a data group from persistent storage within reach of the functional process requiring it. It is considered to include certain associated data manipulation sub-processes necessary to achieve the Read.
- A *Write* (W) is a data movement that moves a data group lying inside a functional process to persistent storage. It is considered to include certain associated data manipulation sub-processes necessary to achieve the Write.

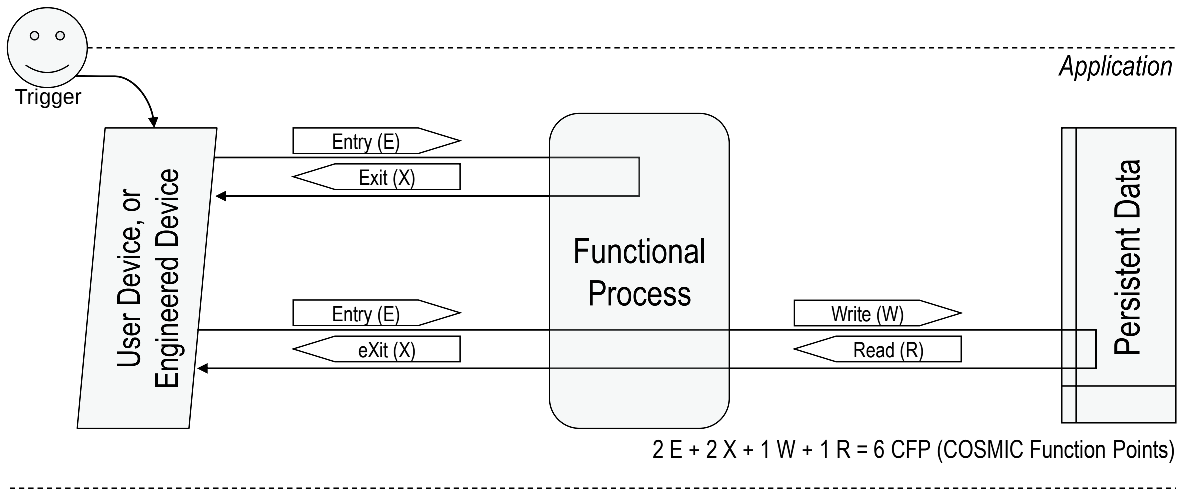

Illustration 1: COSMIC Functional Sizing Principle

The total functional size is an aggregation of all data movements between data groups, functional processes and user devices, or engineered devices within the boundary of the software.

1.2 Using UML 2.0 Sequence Diagrams in Agile

The concept of data movement is very close to the concept of messages between entities – classes, objects, data stores – in UML™ [21]. Data groups, in turn, resemble objects. Thus, why not counting software based on *Sequence Diagrams*? Sequence diagrams are an UML™ notation that is understood for communicating dynamic behaviour of software among stakeholders. According to Donald Bell [6], "the sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur. …An organization's business staff can find sequence diagrams useful to communicate how the business currently works by showing how various business objects interact."

Drafting sequence diagrams is mainly a matter of available media. In most cases, whiteboards and walls simply are not big enough, and good design tools are expensive and not easily available. Scott W. Ambler recommends using sequence diagrams foremost of all other UML artefacts in agile software development teams [4]. If available, the team appreciates them for facilitating mutual understanding with the sponsor and the users, and spotting design weaknesses early. Drafting sequence diagrams for software development is a matter of available media and skills. If done, the team appreciates it as a means for facilitating mutual understanding and spotting weaknesses early.

Sequence diagrams allow the development team to communicate effectively with business people, adopting the end-user's viewpoint to look how data is moved through the system. Implementation details can be clarified that do not affect user experience but might affect quality features of the finished software product such as performance. Many teams use them to identify risks for performance bottlenecks, especially in real-time software applications.

In agile development, requirements are stated as *User Stories*. Agile teams use user stories (the counterpart of use cases in iterative development) to reflect the customer's needs in a convenient way. Thus, when creating the sequence diagram for a user story, some of the tasks needed for implementing the user story become immediately apparent, see section 1.6: Sequence Diagramming in User Stories. We call tasks derived from user stories a *Story Point*. The functional story points correspond to the messages – or data movements – that are exchanged among the entities, or data groups. Thus the effort to create a sequence diagram is awarded not only by a better common understanding of the story, but also by an indisputable decomposition of an user story into story points, at least for the functional part.

For the non-functional part of user story implementation, the sequence diagrams also give hints, e.g., for identifying critical entities for performance, such as data sinks being overwhelmed with data for storing after some high-volume data movements.

1.3 Sizing Sequence Diagrams

UML™ 2.0 sequence diagrams identify class objects that occur as data groups in functional processes, and messages that implement data movements. Levesque et. al. [18] propose that each use case is a functional process. The actors of the use case are the users. The entities of the sequence diagram are the data groups. The data movements correspond to the messages among the entities of the sequence diagram. Levesque et. al. claim that results of this approach compared to expert counts exhibit a difference of 8% only. See Marín [19] for comparisons with various other sizing methods for conceptual models.

Identifying data movements between data groups according the ISO/IEC 19761 international standard is a rather small additional effort when drafting sequence diagrams. However, not all classes that need being implemented appear in functional processes, because some classes do not directly implement FURs but rather respond to technical requirements, representing viewpoints usually not shared with users. The distinction between classes appearing in functional processes, and “technical” entities, is not a problem in practice; regarding the theory, the COSMIC reference material provides suitable guidance [1].

1.4 The Four Rules for Identifying Data Movements in a Count

When practising functional sizing of sequence diagrams with software engineers having a limited exposure to functional sizing methods, and not being certified as function point specialists, the following easy-to-learn characterisation rules for identifying data movements among the arrows in sequence diagrams are sufficient:

1. Does the data movement implement a set of FURs?
2. Is it triggered by an event?
3. Does the triggering event occur outside the boundary of the software?
4. Does the functional process execute all that is required to be done in response to the triggering event?

These rules prevent the team from digging into too much detail when creating sequence diagrams and counting message arrows. Even performed manually, it requires only a small additional effort for developers when sequence diagrams are available. This effort pays off for adding clarity and facilitating effort estimations. Using these four criteria eventually provide higher adherence to the ISO/IEC 19761 standard as measured by Levesque; however, measurements aren’t available yet.

1.5 Embracing Change with User Stories

Scott W. Ambler points out that “agile teams will identify new stories during construction iterations, split existing stories when you realise that they’re too large to be implemented in single iteration, re-prioritise existing stories, or remove stories that are no longer considered to be in scope. The point is that stories evolve over time just like other types of requirements models evolve.” Further enhancement requests may be identified by sales and support people during the production phase and then forwarded to a development team as they are working on an upcoming release [4].

1.6 Sequence Diagramming in User Stories

In Agile, the recommended practice is drawing sequence diagrams [19] on the back – or attached – to user story cards, as Grant Rule proposed in a recent note, see [24]. Sequence diagrams create a common understanding and allow identifying risks with the planned software at an early stage[1].

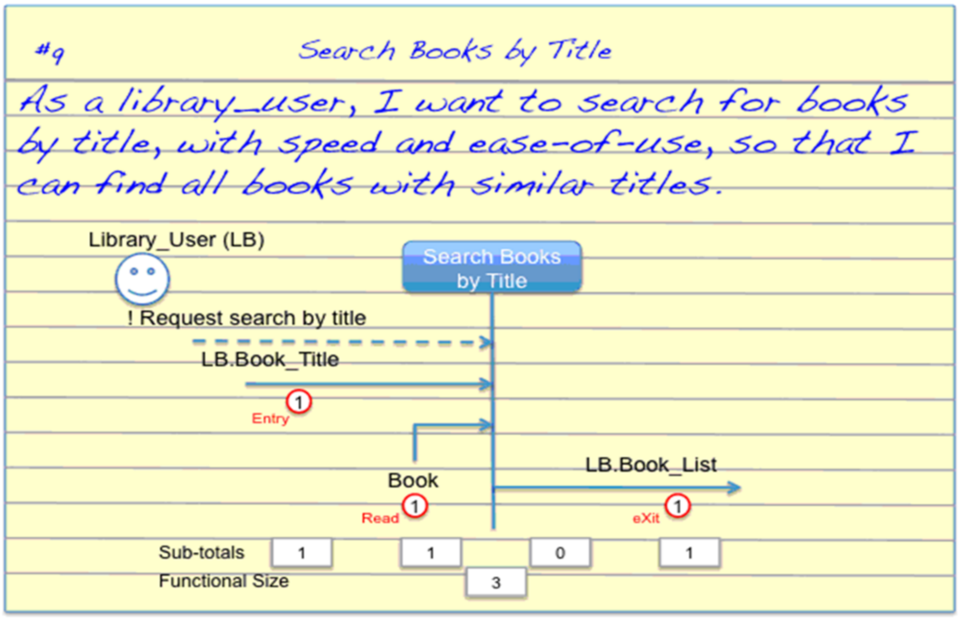
#9
Search Books by Title
As a library_user, I want to search for books by title, with speed and ease-of-use, so that I can find all books with similar titles.
Library_User (LB)
Search Books by Title
! Request search by title
LB.Book_Title
1
Entry
Book
1
Read
LB.Book_List
1
eXit
Sub-totals
1
1
0
1
Functional Size
3

Illustration 2: Sample user story with FSM according ISO/IEC 19761

The change needed to integrate functional size measurement into user stories is to replace the usual template for writing user story cards.

In general the user story template looks as [25]:

> As a … [stakeholder role] …
> I want to … [perform an action / record some information] …
> [With some frequency and/or quality characteristic] …
> So that … [description of value or benefit is achieved].

[1] This sample user story card is acknowledged to the creators Grant Rule (Software Measurement Services Ltd) and Peter Fagg (Pentad Ltd).

When adapting it to ISO/IEC 19761, it reads according Grant Rule 15:

> As a … [functional user] …
> I want to … [respond to an event / retrieve some data] …
> [With some frequency and/or quality characteristic] …
> So that … [description of value or benefit is achieved].

With just these few adaptation to the international standard, agile teams adopt a standardized and comparable size measurement method that yield identical results across time, across teams, across projects, and across organizations. Developers, product owners, and their customers thus have a low-cost, effective tool to use when estimating, tracking progress, and assessing value-for-money.

Each of the lines in a user story may give rise to one or more *Story points*, piece in the user story that make up for functional or non-functional user requirements. In the sample case of Illustration 2, we have three elementary data movements: 1) Enter book title, 2) Read from data store containing existing books, and 3) Exit with book list, and four quality-related tasks: 4) Include subtitle search, 5) Speedy search, 6) Forgiving grammar checks, and 7) Pattern matching search. All these story points impact the amount of work needed to implement the full user story.

1.7 Conjecture

As a preliminary conclusion, a single solution seems not yet to be available. Our conjecture is, this is because if people want to size any sequence diagram, they probably count diagrams with unstandardized granularity.

To illustrate the conjecture, we present a sample COSMIC counting case.

2 A Sample Case

2.1 A Transportation Service Helpdesk Story

Assume a transportation company – railway or airline – wants to enhance their helpdesk operations and make them fit for today's social media environment. To start with, they consider the following five user stories for implementation:

[SW-1] As a helpdesk staff I want to identify a client without having to ask him who he is and get his credentials, regardless whether he calls by phone, e-mail, or contacts me by chat in a social networking environment, such that I can charge any service fees or ticket sales directly to his or her telecommunications or credit card bill.

[SW-2] As a registered customer of the travel company, I expect that the helpdesk will recognize me on the basis of the SIM card in my mobile phone, so I can pay my travel ticket in accordance with my previously recorded user profile either via credit card or telecommunication bill.

[SW-3] As a non-registered customer, I want to be able to identify me quickly and easily with my credit card and my mobile phone to use the services of my travel service provider immediately.

[SW-4] As a socially committed person I would like to plan, book and amend my travels on short notice, and cancel, at any time day or night maybe, in order to be where I have just the most fun or best work to do.

[SW-5] As a user of e-mail on a computer or smartphone I want to store my SIM certificate that I need for authentication in the usual certificate store provided by my operating system, so that I can sign my e-mail when contacting the helpdesk and identify myself as easily as when calling via mobile phone or using the smartphone.

These five user story are significantly more complicated than the one in section 1.6. They do not as easily reveal the functional story points that would allow transforming the user stories into sequence diagrams. Before writing the sequence diagrams, the application architecture must be designed and the boundaries identified. Moreover, user story [SW-4] is not connected to any functionality at all. This is a distributed environment with many services involved such as telecommunication companies offering authentication services based on a SIM card, and credit card institutes that offer the possibility to validate a credit card used for buying travelling services, e.g., tickets. Thus the first thing to do is to design the application architecture.

2.2 SIM Card Identification

The basic idea that the solution architects pursue is based on identifying customers automatically using their SIM Card. The ***S**ubscriber **I**dentity **M**odule* (SIM) that everyone carries in the mobile phone can be used to identify customers, since when they call, they provide their mobile number (hiding it being discouraged) or at least they use a *Smart-phone* or *Mobile Phone* that is connected to their communication device, and the identity of the SIM card can be validated. In its simplest version, credit card issuer uses this to verify the user by sending a validation code via SMS on the credit card holder's mobile phone. However, even if they use the SIM card to contact the customer, they usually cannot get the SIM Card ID. Such identification requires the cooperation of the telecommunication operator that issued the SIM card.

For our fictive case, we assume they offer a new service to provide authentication that the transportation company can use for identifying rapidly customers when they call in.[1] While it works when customers are calling by mobile phone and

[1] Obviously, the transportation company needs the approval of the customer to use such services from the telecommunication company, for privacy and data protection reason.

chatting over smartphones, it needs an additional service when using e-Mail, namely providing a certificate that encrypts the SIM card information with each e-Mail sent to the transportation helpdesk. Such a certificate can be created when initially subscribing to the helpdesk service or on the fly if the mobile phone is connected. – In the following, we use the term 'caller' even if communication uses other services than voice, e.g., chat or e-Mail.

2.3 Decomposing into Application Layers

The top layer is the helpdesk application that has at least three elementary processes

- enrolling to the helpdesk service with identification and authentication by the telecommunication operator and the credit card institute;
- maintaining personal profile information such as addresses and SIM cards;
- the helpdesk ticket system that starts with the caller calling in, and ends with giving advice against payment, re-routing the traveller who missed a connection, or executing a business transaction such as a ticket acquisition.

Callers use phone, e-mail, or chat to communicate with the helpdesk.

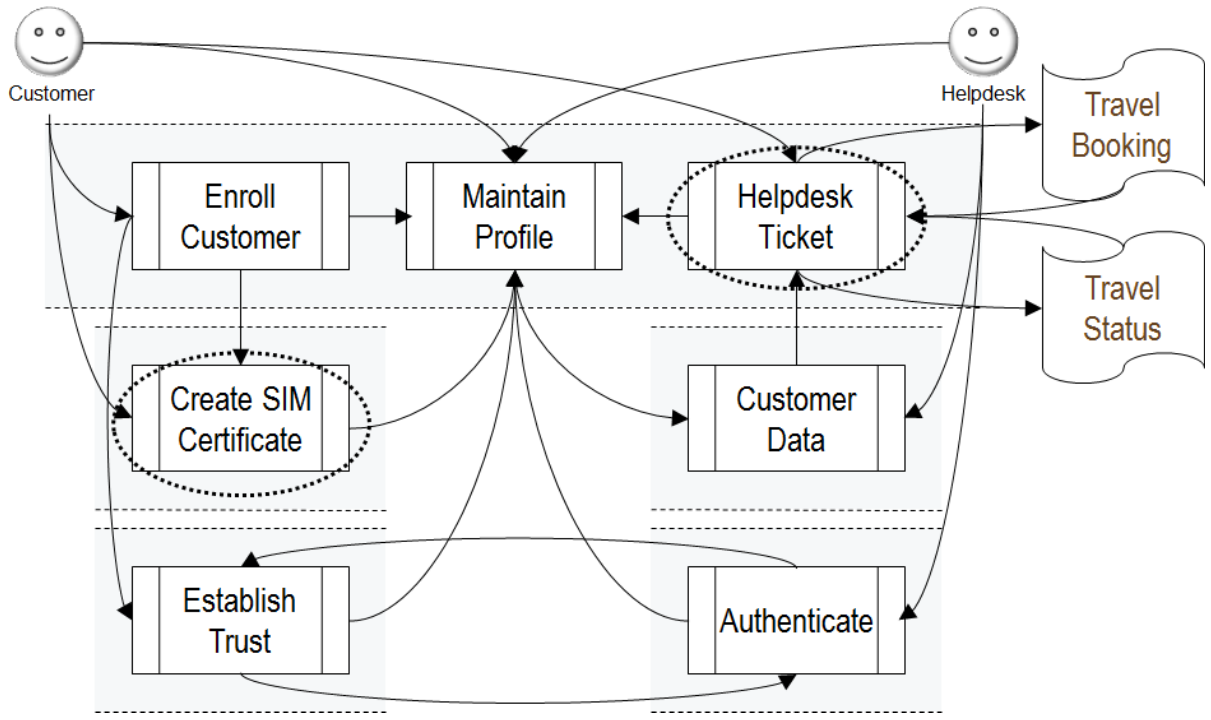

Illustration 3: Application Overview

Besides the top layer, four more application layers are needed:

- a local application running on the equipment used for e-mailing, such as a laptop, touch-pad, or smartphone, for creating SIM certificates.
- to present the customer data from the CRM system to the helpdesk people;

- to identify and authenticate the customer by the telecommunication operator based on his or her subscription;
- to establish the trust network used for identifying callers as new customers, involving telecommunication operator and the bank.

Furthermore, the helpdesk team uses travel booking and travel status and tracking systems for inquiries and ticketing; these systems are not considered containing objects of interest for our case study because they already exist and possibly don't need any change. It might be when information is moved, say, from a real time train position system to a chat or e-Mail, an interface is needed for reading and writing data, then we count these data movements only but not the other layers involved are within the scope of this count.

2.4 Sizing based on Sequence Diagrams

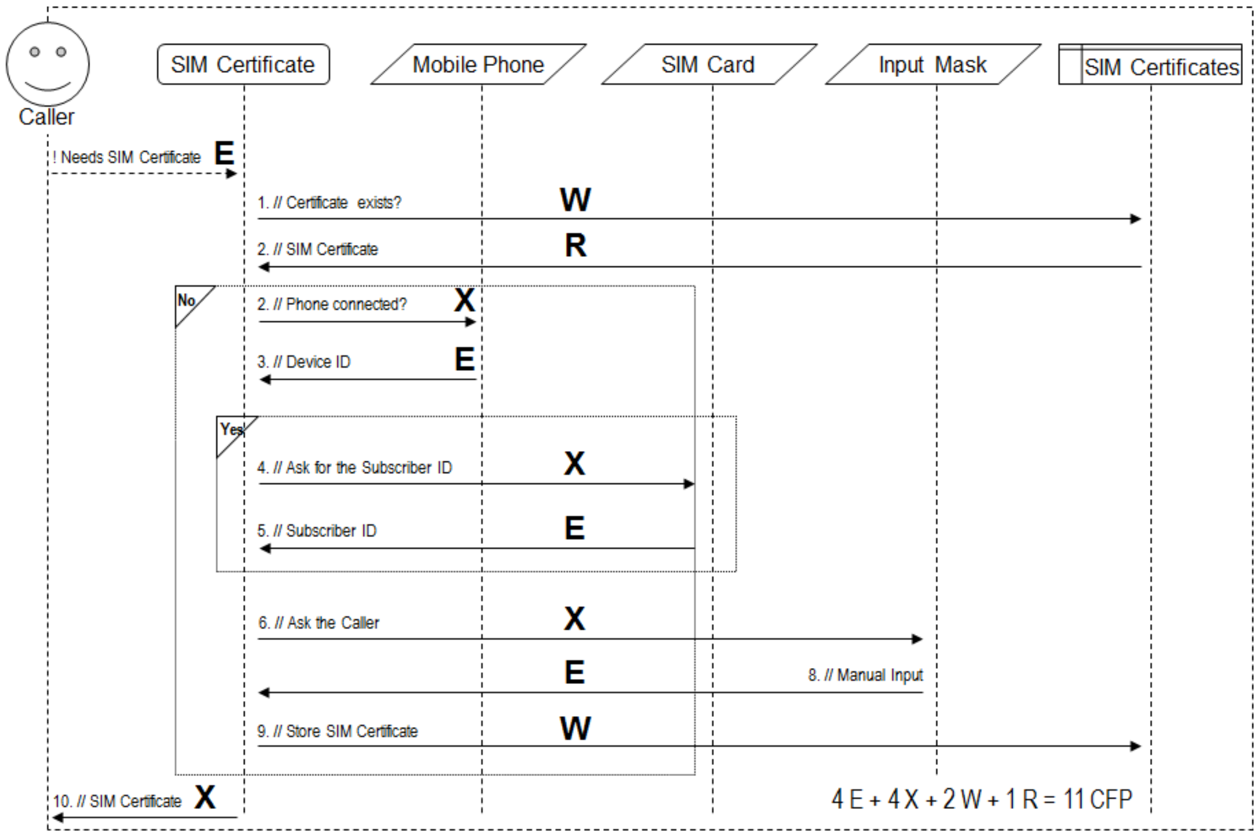

Illustration 4: Create SIM Card - Application Layer

As examples, we consider the first elementary process in the second layer “Create SIM Certificate”, and the third elementary process in our top layer, the “Helpdesk Ticket” issue resolution ticketing application. When practising functional sizing of sequence diagrams with software engineers having a limited exposure to functional sizing methods, the four rules in section 1.4 prevent the team from digging into too much detail when creating sequence diagrams; the COSMIC rules standardize their granularity.

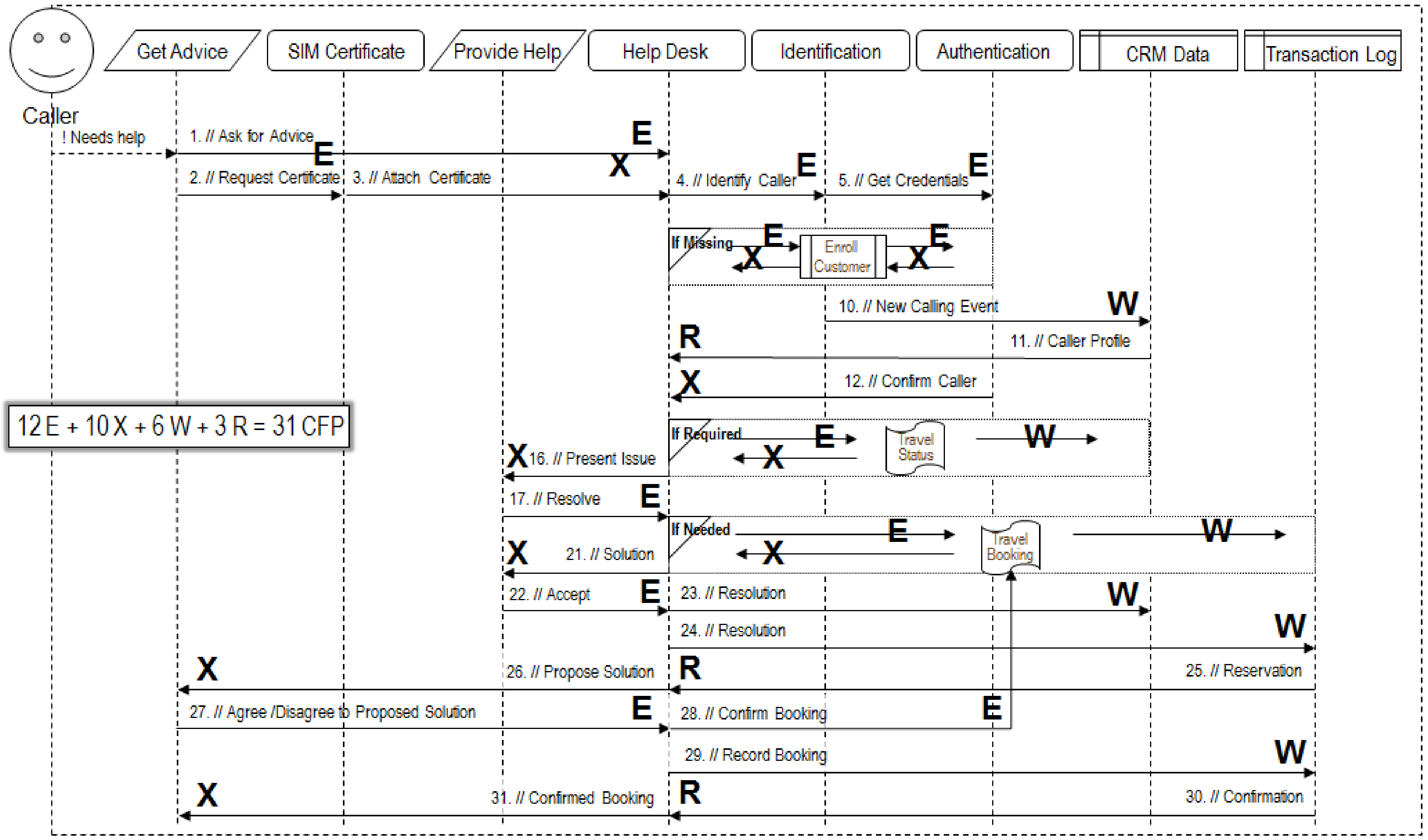

Illustration 5: Ticketing Application Layer

The objects in the sequence diagrams correspond to the COSMIC user devices, functional processes, and data groups. The distinction between these is of less importance when looking after sequence diagrams; all correspond to objects of interest. For the purpose of counting, their forms match those in Illustration 1: COSMIC Functional Sizing Principle

Counting message arrows, even performed manually, requires only a small additional effort for developers when sequence diagrams are available. This effort pays off for adding clarity and facilitating effort estimations. Using these four criteria eventually provide higher adherence to the ISO/IEC 19761 standard as measured by Levesque; however, measurements aren't available yet.

In the second case presented in Illustration 5, all messages correspond to a data movement. Not shown are calculations and decisions taken within a data object, creating a self-closed loop, that normally also appear in a sequence diagram. Entries to, and eXits from, other elementary processes or application layers are shown within dotted boxes, where the data movements have not been labelled. The total amounts to 31 CFP (COSMIC Function Points).

Note the data movements between the help desk process, and the interface to the help desk people ("Provide Help"). Although functionality is rather simple, the quality of the help desk process very much depends from ergonomic features of that human interface, not reflected in this COSMIC count.

2.5 Findings

When counting messages that meet the four requirements given in Section 9, and limiting to messages between two different data objects, we see no difference between counting messages and counting data movements.

It is not obvious whether all persistent data and all devices are represented as data objects in sequence diagrams, or in contrary, whether data objects exist that are neither persistent data nor devices. A typical such entity would be a temporary storage – for instance used to speed up execution.

However, when the four conditions are applied duly, there is no way such a data object ever receives a message. Thus it's not needed in a sequence diagram.

3 Application to the Software Development Process

3.1 Counting Size in Real Time

Even if not agile, software development processes adapt flexibly to the knowledge acquired and customer's requirements change that normally affect the development process. Following Ambler [4], sequence diagramming suits best as a communication tool within the development team and with the customer. This is valid for traditional software development approaches as well.

Structure diagramming allows keeping a continuously updated size count with identical viewpoint and stable granularity during the full software development process, enabling for scope management of the development project. Any time during development, it is made clear what is being developed and agreement can be negotiated.

3.2 The Buglione-Trudel Matrix

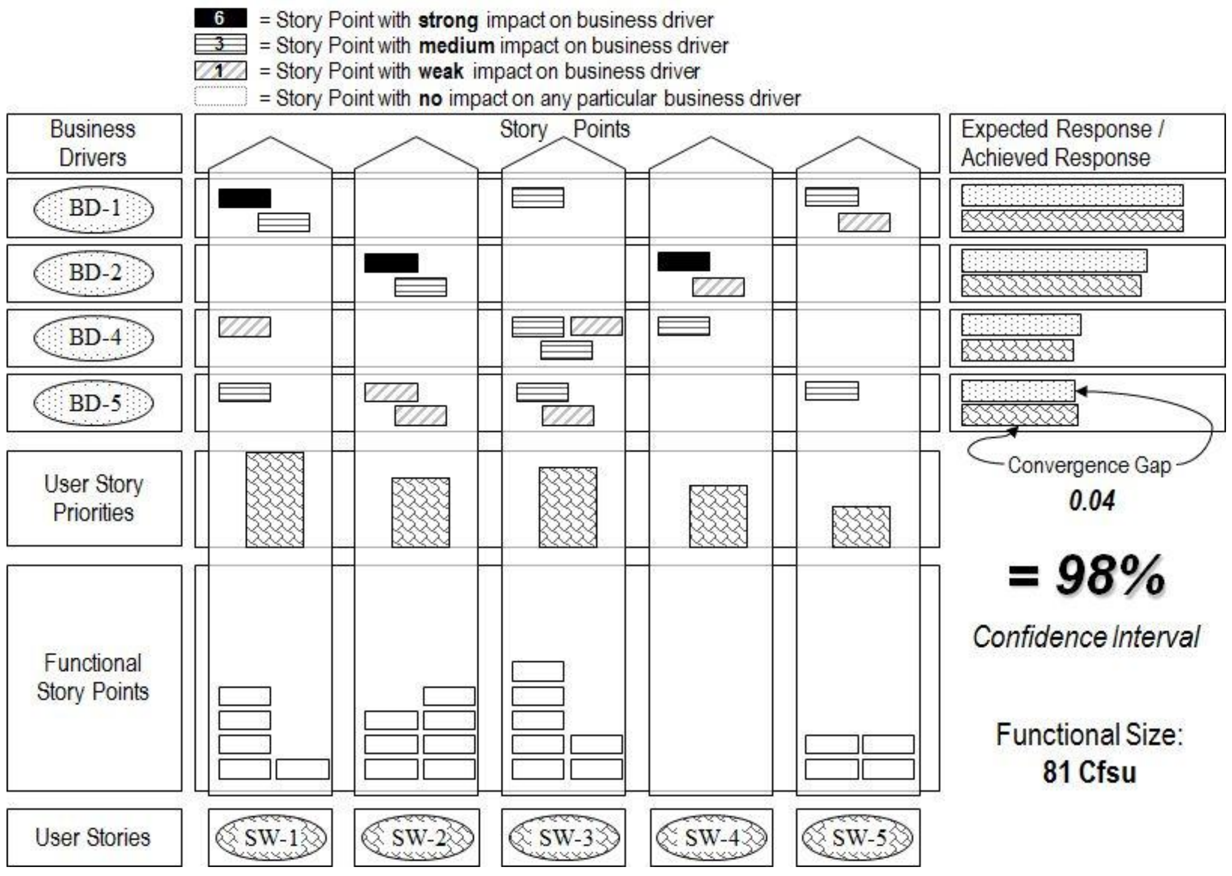
6 = Story Point with strong impact on business driver
3 = Story Point with medium impact on business driver
1 = Story Point with weak impact on business driver
= Story Point with no impact on any particular business driver
Business Drivers
Story Points
Expected Response / Achieved Response
BD-1
BD-2
BD-4
BD-5
User Story Priorities
Functional Story Points
User Stories
SW-1
SW-2
SW-3
SW-4
SW-5
Convergence Gap
0.04
= 98%
Confidence Interval
Functional Size:
81 Cfsu

Illustration 6: Sample Buglione-Trudel Matrix for Helpdesk Software Development

The Buglione-Trudel matrix has been proposed [15] to organize project information when developing software. It is primarily intended for agile teams where the knowledge of developers is being harvested for uncovering the optimum development approach, based on story points; however, it works for all kind of projects.

Story points are the functional or non-functional pieces that together make up for a user story. Functional and non-functional story points, taken from well-formed user stories, are separated and matched against *Business Drivers*, as introduced by Denney [12] for successfully managing software projects.

When implementing software that supports the help desk, meeting these business drivers is conclusive for meeting customer's needs. Since BD-3: Friendliness is not something that software can impact (except the software is buggy, what we exclude), we focus on BD-1, BD-2, BD-4, and BD5 as software development business drivers, and use the Buglione-Trudel matrix to associate story points with those goals.

	Topics	*Attributes*
BD-1	Responsiveness	No waiting loops – immediate response
BD-2	Compelling	Can commit for connections and reimbursement
BD-3	Friendliness	Keeps cool under stress – calm down
BD-4	Personal	Knows frequent travellers from login or mobile
BD-5	Competence	Able to solve traveller’s problem

Table 1: Business Drivers for Motivating Travellers to use Help Desk

In many case, non-functional story points impact the business drivers much more than the functional items. Thus successful project managers plan for, or let developers choose, to implement such story points only if they have impact on the business drivers. For functional story points, the functional size they carry is more decisive whether to implement them. For more details on using the Buglione-Trudel matrix for agile teams see [15].

4 Conclusion

The COSMIC measurement rules add an important feature to UML sequence diagrams: an international standard that standardizes granularity. UML does not allow predicting whether there is sufficient detail or not in any of its artefacts; and this might be even the reason why sequence diagrams are not as popular as they ought to be. This is definitely the reason why Use Case Points cannot be used for benchmarks. Use Case Points count the granularity of the Use Case analysis only; nothing else. Also, when drafting sequence diagrams, there is a danger that more and more details pop up and the developer never finishes the draft.

Based on the COSMIC measurement manual, there are rules for decomposition and layers of software. Both help developers to identify the scope of their sequence diagrams, and if they do so, drafting the diagram and sizing with COSMIC go hand-in-hand.

Thus sizing with COSMIC neatly fits into agile development practice, and enhances agile project management thanks to the connection of sequence diagrams with the COSMIC functional size measurement practice. COSMIC links software engineering practise to software project management.

Thus, the original question whether sequence diagrams can be counted using COSMIC probably is the wrong question. It's the other way round: COSMIC helps teams to benefit from the full power of sequence diagramming.

References

[1] Abran, A. et. al.: The COSMIC Functional Size Measurement Method – Version 3.0.1 – Measurement Manual, http://www.cosmicon.com/ [seen 2-May-2011] (2009)

[2] Aguanno, K. (ed.): Managing Agile Projects, Multi-Media Publications, Lakefield, Ontario, CA (2004)

[3] Akao, Y. et. al.: Quality Function Deployment, Productivity Press, University Park, IL (1990)

[4] Ambler, S.W.: The Object Primer, 3rd Edition – Agile Model–Driven Development With UML 2.0, Cambridge University Press, New York, NY (2004)

[5] Beck, K.: eXtreme Programming Explained, Addison-Wesley, Boston (2000)

[6] Bell, D.: UML basics: The Sequence Diagram – Introductory Level. In: IBM Developer Works, http://www.ibm.com/developerworks/rational/library/3101.html [seen 2-May-2011] (2009)

[7] Buglione, L., Trudel, S.: Guideline for Sizing Agile Projects with COSMIC. In: Proceedings of the IWSM / MetriKon / Mensura 2010, Stuttgart, Germany (2010)

[8] Bundschuh, M., Dekkers. C.: The IT Measurement Compendium: Estimating and Benchmarking Success with Functional Size Measurement, Springer Verlag Berlin – Heidelberg (2008)

[9] Hill, P. ed., Practical Software Project Estimation 3rd Edition, McGraw-Hill, New York, NY (2010)

[10] Cohn, M.: Agile Estimating and Planning, Prentice Hall, New Jersey (2005)

[11] Creveling, C.M., Slutsky, J.L., Antis, D.: Design for Six Sigma, Prentice Hall, NJ (2003)

[12] Denney, R.: Succeeding with Use Cases – Working Smart to Deliver Quality, Booch–Jacobson–Rumbaugh – Series, Addison-Wesley, New York, NY (2005)

[13] Fehlmann, Th.: The Impact of Linear Algebra on QFD, in: International Journal of Quality & Reliability Management, Vol. 21 No. 9, pp. 83-96, Emerald Group Publishing Ltd., Bradford, UK (2005)

[14] Fehlmann, Th.: Defect Density Prediction with Six Sigma, in: Proceedings of the 6th Software Measurement European Forum, Rome, Italy (2009)

[15] Fehlmann, Th.: Agile Software Projects with Six Sigma. 3rd European Research Conference on Continuous Improvement and Lean Six Sigma, Glasgow (2011)

[16] Fenton, N.E., Neil, M., and Marquez, D., Using Bayesian Networks to Predict Software Defects and Reliability. Proceedings of the Institution of Mechanical Engineers, Part O, Journal of Risk and Reliability: p. 701-712 (2008).

[17] Fuqua, A.M.: Using Function Points in XP – Considerations. In: Extreme Programming and Agile Processes in Software Engineering, LNCS vol. 2675/2003, pp. 1013ff (2003)

[18] Levesque, G., Bevo, V. Cao, D.T.: Estimating Software Size with UML Models. In: Proceedings of the 2008 C^3S^2E Conference, Montreal, pp. 81--87 (2008)

[19] Marín, B., Giachetti, G., Pastor, O.: Measurement of Functional Size in Conceptual Models: A Survey of Measurement Procedures Based on COSMIC. In: Dumke et. al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 170--183 (2008)

[20] Miller, G. A.: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information, The Psychological Review, Vol. 63, 1956, pp. 81-97 (1956)

[21] OMG Unified Modeling Language™ (OMG UML), Superstructure V2.2 pp. 506--524, http://www.omg.org/spec/UML/2.2/Superstructure/PDF/ [seen 2-May-2011] (2009)

[22] Poppendieck, M.& T.: Implementing Lean Software Development, Addison-Wesley, New York, NY (2007)

[23] Prasetya, W.: JUnit 4.x Quick Tutorial, linked to JUnit.org developer community site: In http://code.google.com/p/t2framework/wiki/JUnitQuickTutorial. Seen 2-May-2011 (2009)

[24] Rule, G.: Sizing User Stories with the COSMIC FSM method, SMS Exemplar, http://www.smsexemplar.com/wp-content/uploads/20100408-COSMICstories-article-v0c1.pdf. Seen 2-May-2011] (2010)

[25] Schwaber, K., Beedle, M.: Agile Software Development with Scrum, Prentice Hall (2008)